EL764085657

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Multiple User Authentication for Online Console-Based Gaming

Inventor(s):

Darren Anderson
Tony Chen
Boyd Multerer

ATTORNEY'S DOCKET NO. MS1-766US

TECHNICAL FIELD

This invention relates to console-based gaming systems, and more particularly, to methods for authenticating multiple identities in a single reply/request exchange between a game console and an authentication entity.

BACKGROUND

Traditionally, gaming systems with a dedicated console were standalone machines that accommodated a limited number of players (e.g., 4). PC-based gaming grew in popularity in part due to the ability to play games online with many remote players over a network (e.g., the Internet). Thus, one trend for dedicated gaming systems is to provide broadband capabilities to facilitate online gaming. Microsoft Corporation recently announced its Xbox™ video gaming system that is equipped with a hard disk drive to enhance gaming, and broadband connectivity to support online gaming.

Creating an online gaming system for a dedicated console poses several unique and difficult problems. One problem concerns authentication of the participants. To establish an online gaming event, a local game being played on one game console goes online and communicates with other game consoles, players, and/or online services. This involves some level of trust among the participants, which the game attempts to establish by identifying itself, the game console, and the one or more players currently on the machine to other participants on the network in a secure manner. Additionally, the game console may also want to discover trusted services with which it can communicate over the network.

The PC-based games do not experience such problems. For instance, PC-based games do not typically experience multiple simultaneous users; rather only a

single user is involved in the online game. On a PC, the users can easily enter their data via keyboard and the trusted services are easily configurable. Also, PC users tolerate network operations that take a little longer. If the PC game takes five extra seconds to start because it is making multiple round-trips to an authentication server, no one will complain. This is not the case in the gaming world.

Accordingly, the constraints on a dedicated game console make authentication a difficult problem for the following reasons:

- Consoles do not have keyboards. Game controllers are not efficient data-entry devices, thus user-entered data should be kept to a minimum.
- Gaming systems are plug-and-go; they plug into the wall and are ready for play. Configuration is not expected or tolerated in the game console community.
- Console games should be playable with as little start up time as possible. Players expect to put a game disk into the console, turn it on, and be playing the game a few seconds later.
- Consoles are a closed development environment in order to ensure high content quality. Thus, consoles need to know that they are communicating with trusted, quality controlled services. The need for trusted communications is further driven by the addition of a hard disk drive into the console in that any malicious damage rendered to the hard disk drive's data by an external source makes it difficult or impossible to repair without reformatting.

Cheating is not yet a major problem in the online console-based gaming community. In anticipation that it might one day pose a problem, there is a need for a solution that addresses cheating. Part of the solution is to make the console itself as secure as possible and tamper resistant, such that any tampering by a user will be discovered or render the console inoperable. While security and tamper resistant solutions help, it does not prevent the case where a rogue player writes PC software to emulate a console machine on the network, enabling cheating without a game console.

To prevent such impostor cheating, there is a need for an authentication protocol that verifies a player claiming to be on a console machine really is that player, as well as guarantees that the game console is indeed a trusted game console and not an impostor or one that has been compromised.

SUMMARY

A console-based multi-user authentication process allows multiple users of a game console to be authenticated together in a single request/reply exchange with an authentication entity.

In the described implementation, the game console is equipped with a hard disk drive, a portable media drive, and broadband connectivity to enable network access to a ticket issuing entity and one or more online services. When the game console desires to use the online service, it first obtains a ticket for that service from the ticket issuing entity. The game console submits a request to the ticket issuing entity that contains a game console identity, the identities of the multiple users, and an identity of the desired online service.

In response, the ticket issuing entity generates a ticket containing the game console identity and the multiple user identities together encrypted with the online service's key. The ticket issuing entity returns the ticket to the game console, which passes it onto the online service. The online service uses the ticket to verify the authenticity of the game console and the multiple users. In this manner, the single ticket obtained from the ticket issuing entity proves the particular game console as well as the multiple user identities playing at the game console.

The game console identity is created when the game console first establishes a game account for online gaming. During manufacturing, the game console is constructed with pieces of information that may be made available programmatically (e.g., hard disk ID, CPU ID, serial number, random number, a value derived as a function of an ID or serial number, a quantity or some other mark written onto the hard drive, etc.). This information is recorded in a database, which is subsequently made available to an authentication server. When the game console seeks a game account, it submits the pieces of information to the authentication server. Using the database, the server evaluates whether the pieces are legitimate and correspond to a game console that has not yet established an account. If the evaluation proves positive, a game account is created for that game console and the game console identity is assigned to the game console.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a gaming system with a game console and one or more controllers.

Fig. 2 is a block diagram of the gaming system.

Fig. 3 illustrates a network gaming system in which the Fig. 1 gaming system is connected via a network to other consoles, services, and a ticket issuing entity.

Fig. 4 illustrates a multi-user authentication process involving three participants: a game console, a ticket issuing entity, and an online service.

Fig. 5 is a flow diagram of the multi-user authentication process that facilitates authentication of multiple identities—game console, game title, multiple users.

Fig. 6 is a flow diagram of a process for establishing a game console identity that is used in the multi-user authentication process.

DETAILED DESCRIPTION

The following discussion is directed to console-based online gaming systems and techniques for authenticating multiple identities—game console, game title, multiple users—in one authentication roundtrip. The discussion assumes that the reader is familiar with basic cryptography principles, such as encryption, decryption, authentication, hashing, and digital signatures. For a basic introduction to cryptography, the reader is directed to a text written by Bruce Schneier and entitled, "Applied Cryptography: Protocols, Algorithms, and Source Code in C," published by John Wiley & Sons, copyright 1994 (second edition 1996), which is hereby incorporated by reference.

Gaming System

Fig. 1 shows an exemplary gaming system 100. It includes a game console 102 and up to four controllers, as represented by controllers 104(1) and 104(2).

The game console 102 is equipped with an internal hard disk drive and a portable media drive 106 that supports various forms of portable storage media as represented by optical storage disc 108. Examples of suitable portable storage media include DVD, CD-ROM, game discs, game cartridges, and so forth.

The game console 102 has four slots 110 on its front face to support up to four controllers, although the number and arrangement of slots may be modified. A power button 112 and an eject button 114 are also positioned on the front face of the game console 102. The power button 112 switches power to the game console and the eject button 114 alternately opens and closes a tray of the portable media drive 106 to allow insertion and extraction of the storage disc 108.

The game console 102 connects to a television or other display (not shown) via A/V interfacing cables 120. A power cable 122 provides power to the game console. The game console 102 may further be configured with broadband capabilities, as represented by the cable or modem connector 124 to facilitate access to a network, such as the Internet.

Each controller 104 is coupled to the game console 102 via a wire or wireless interface. In the illustrated implementation, the controllers are USB (Universal Serial Bus) compatible and are connected to the console 102 via serial cables 130. The controller 102 may be equipped with any of a wide variety of user interaction mechanisms. As illustrated in Fig. 1, each controller 104 is equipped with two thumbsticks 132(1) and 132(2), a D-pad 134, buttons 136, and two triggers 138. These mechanisms are merely representative, and other known gaming mechanisms may be substituted for or added to those shown in Fig. 1.

A memory unit (MU) 140 may be inserted into the controller 104 to provide additional and portable storage. Portable memory units enable users to

store game parameters and port them for play on other consoles. In the described implementation, each controller is configured to accommodate two memory units 140, although more or less than two units may be employed in other implementations.

The gaming system 100 is capable of playing, for example, games, music, and videos. With the different storage offerings, titles can be played from the hard disk drive or the portable medium 108 in drive 106, from an online source, or from a memory unit 140. A sample of what the gaming system 100 is capable of playing back include:

1. Game titles played from CD and DVD discs, from the hard disk drive, or from an online source.
2. Digital music played from a CD in the portable media drive 106, from a file on the hard disk drive (e.g., Windows Media Audio (WMA) format), or from online streaming sources.
3. Digital audio/video played from a DVD disc in the portable media drive 106, from a file on the hard disk drive (e.g., Active Streaming Format), or from online streaming sources.

Fig. 2 shows functional components of the gaming system 100 in more detail. The game console 102 has a central processing unit (CPU) 200 and a memory controller 202 that facilitates processor access to various types of memory, including a flash ROM (Read Only Memory) 204, a RAM (Random Access Memory) 206, a hard disk drive 208, and the portable media drive 106. The CPU 200 is equipped with a level 1 cache 210 and a level 2 cache 212 to

temporarily store data and hence reduce the number of memory access cycles, thereby improving processing speed and throughput.

The CPU 200, memory controller 202, and various memory devices are interconnected via one or more buses, including serial and parallel buses, a memory bus, a peripheral bus, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, and a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus.

As one suitable implementation, the CPU 200, memory controller 202, ROM 204, and RAM 206 are integrated onto a common module 214. In this implementation, ROM 204 is configured as a flash ROM that is connected to the memory controller 202 via a PCI (Peripheral Component Interconnect) bus and a ROM bus (neither of which are shown). RAM 206 is configured as multiple DDR SDRAM (Double Data Rate Synchronous Dynamic RAM) that are independently controlled by the memory controller 202 via separate buses (not shown). The hard disk drive 208 and portable media drive 106 are connected to the memory controller via the PCI bus and an ATA (AT Attachment) bus 216.

A 3D graphics processing unit 220 and a video encoder 222 form a video processing pipeline for high speed and high resolution graphics processing. Data is carried from the graphics processing unit 220 to the video encoder 222 via a digital video bus (not shown). An audio processing unit 224 and an audio codec (coder/decoder) 226 form a corresponding audio processing pipeline with high fidelity and stereo processing. Audio data is carried between the audio processing

unit 224 and the audio codec 226 via a communication link (not shown). The video and audio processing pipelines output data to an A/V (audio/video) port 228 for transmission to the television or other display. In the illustrated implementation, the video and audio processing components 220-228 are mounted on the module 214.

Also implemented on the module 214 are a USB host controller 230 and a network interface 232. The USB host controller 230 is coupled to the CPU 200 and the memory controller 202 via a bus (e.g., PCI bus) and serves as host for the peripheral controllers 104(1)-104(4). The network interface 232 provides access to a network (e.g., Internet, home network, etc.) and may be any of a wide variety of various wire or wireless interface components including an Ethernet card, a modem, a Bluetooth module, a cable modem, and the like.

The game console 102 has two dual controller support subassemblies 240(1) and 240(2), with each subassembly supporting two game controllers 104(1)-104(4). A front panel I/O subassembly 242 supports the functionality of the power button 112 and the eject button 114, as well as any LEDs (light emitting diodes) or other indicators exposed on the outer surface of the game console. The subassemblies 240(1), 240(2), and 242 are coupled to the module 214 via one or more cable assemblies 244.

Eight memory units 140(1)-140(8) are illustrated as being connectable to the four controllers 104(1)-104(4), i.e., two memory units for each controller. Each memory unit 140 offers additional storage on which games, game parameters, and other data may be stored. When inserted into a controller, the memory unit 140 can be accessed by the memory controller 202.

A system power supply module 250 provides power to the components of the gaming system 100. A fan 252 cools the circuitry within the game console 102.

A console user interface (UI) application 260 is stored on the hard disk drive 208. When the game console is powered on, various portions of the console application 260 are loaded into RAM 206 and/or caches 210, 212 and executed on the CPU 200. The console application 260 presents a graphical user interface that provides a consistent user experience when navigating to different media types available on the game console.

The game console 102 implements a cryptography engine to perform common cryptographic functions, such as encryption, decryption, authentication, digital signing, hashing, and the like. The cryptography engine may be implemented as part of the CPU 200, or in software stored on the hard disk drive 208 that executes on the CPU, so that the CPU is configured to perform the cryptographic functions.

The gaming system 100 may be operated as a standalone system by simply connecting the system to a television or other display. In this standalone mode, the gaming system 100 allows one or more players to play games, watch movies, or listen to music. However, with the integration of broadband connectivity made available through the network interface 232, the gaming system 100 may further be operated as a participant in a larger network gaming community. This network gaming environment is described next.

Network Gaming

Fig. 3 shows an exemplary network gaming environment 300 that interconnects multiple gaming systems 100(1), …, 100(g) via a network 302. The network 302 represents any of a wide variety of data communications networks. It may include public portions (e.g., the Internet) as well as private portions (e.g., a residential Local Area Network (LAN)), as well as combinations of public and private portions. Network 302 may be implemented using any one or more of a wide variety of conventional communications media including both wired and wireless media. Any of a wide variety of communications protocols can be used to communicate data via network 302, including both public and proprietary protocols. Examples of such protocols include TCP/IP, IPX/SPX, NetBEUI, etc.

In addition to gaming systems 100, one or more online services 304(1), …, 304(s) may be accessible via the network 302 to provide various services for the participants, such as hosting online games, serving downloadable music or video files, hosting gaming competitions, serving streaming audio/video files, and the like. The network gaming environment 300 may further involve a key distribution center 306 that plays a role in authenticating individual players and/or gaming systems 100 to one another as well as online services 304. The distribution center 306 distributes keys and service tickets to valid participants that may then be used to form games amongst multiple players or to purchase services from the online services 304.

The network gaming environment 300 introduces another memory source available to individual gaming systems 100—online storage. In addition to the portable storage medium 108, the hard disk drive 208, and the memory unit(s) 140, the gaming system 100(1) can also access data files available at remote

storage locations via the network 302, as exemplified by remote storage 308 at online service 304(s).

Multi-User Authentication

To participate in an online gaming situation, it is desirable for every participant to authenticate themselves to one another. Ideally, the entities that should be authenticated include the users, the game title, the game console, and any online service the might be involved. One approach to authenticating each entity is to employ the well-known Kerberos authentication protocol, which is described in the above referenced Schneier book. With the Kerberos authentication protocol, each user would perform an independent authentication cycle with the key distribution center because Kerberos is only used to authenticate a single user identity at a time. Unfortunately, this results in multiple authentication cycles for the various users, which is undesirable in the gaming context.

To authenticate multiple users, the gaming system implements an authentication process that allows simultaneous authentication of all entities—game console, game title, and multiple users—in one request/reply exchange with the key distribution center. Moreover, a single ticket can be used to prove the particular game console and the multiple user identities playing at the game console. This is very efficient and highly desirable in the gaming context. In the described implementation, the process is a Kerberos-like authentication protocol.

Fig. 4 shows three primary participants in the multi-user authentication process: the gaming system 100(1), an online service 304, and the key distribution center 306. The participants are networked together via the network 302 (not

shown in Fig. 4) and are each capable of performing one or more routine cryptographic functions, such as encryption, decryption, one way hashing, random number generation, digital signing, and the like. Although more participants may be involved in the online gaming event, the illustrated participants represent an exemplary set of participants that are involved in the multi-user authentication process.

For discussion purposes, suppose there are four users of the gaming system 100(1), as represented by the four controllers 104(1)-104(4). Each user is given an identity U_1, U_2, U_3, and U_4 and is assigned a user key K_1, K_2, K_3, and K_4. The game console 102 is also assigned its own identity X and a game console key K_X. (One exemplary approach to assigning the game console identity and key is described below in more detail with reference to Fig. 6.) Additionally, the game title, which is shown here as a game disc 108, is assigned a separate identity G. In a similar manner, the online service 304 is assigned its own identity A and a key K_A.

The multi-user authentication process may be understood as a four-step process conducted in two roundtrip communication cycles. The first two steps occur during a single roundtrip request/reply exchange between the gaming system 100(1) and the key distribution center 306, which is illustrated as paths 402 and 404. The latter two steps occur during a single roundtrip request/reply exchange between the gaming system 100(1) and the online service 304, which is illustrated as paths 406 and 408. Note that the 406 request and 408 response are usually piggy backed on a regular online service request, and thus does not incur an extra round trip. Thus, the full authentication process may be carried out very quickly, with minimal information exchanges between the participants.

During the first request/reply exchange, the gaming system 100(1) submits a request asking the key distribution center 306 to issue a single ticket on behalf of all identities—game console X, game title G, and all four users U_1, U_2, U_3, and U_4 —for purposes of participating with the online service 304 (i.e., path 402 in Fig. 4). The key distribution center 306 generates and returns a ticket for use with the desired online service (i.e., path 404 in Fig. 4). In this manner, the key distribution center 306 functions as a ticket issuing entity that issues tickets for services.

During the second request/reply exchange, the gaming system 100(1) submits the ticket to the online service 304 for authentication (i.e., path 406 in Fig. 4). The ticket is piggyback information along with the first request for the online service, and does not need to be sent in a separate request or until the service is needed. If valid, the online service 304 sends back a reply that can be used by the gaming system 100(1) to authenticate the online service 304 (i.e., path 408 in Fig. 4). If all entities are authenticated, the gaming system 100(1) can trust the returned results from the online service and can continue to interact with the online service 304.

The process illustrated in Fig. 4 is advantageous in that a single ticket obtained from the key distribution center 306 is used to prove the game console and the multiple user identities playing at the game console. This is much more efficient than the traditional approach where multiple tickets are needed, one for each authenticated principal.

Fig. 5 shows the multi-user authentication process 500 that is implemented by the three participants of Fig. 4. The process can be implemented in software as computer-executable instructions stored on various storage media at the

participants. When executed, the instructions direct the various participants to perform operations illustrated as blocks in Fig. 5. The tasks are illustrated beneath headings "Key Distribution Center", "Game Console", and "Online Service" to convey an exemplary location as to which entities are performing them. The multi-user authentication process 500 will be described with reference to both Figs. 4 and 5.

At block 502 in Fig. 5, the game console 102 generates validated user identities based on the user identities U_1, U_2, U_3, and U_4 and user keys K_1, K_2, K_3, and K_4. More specifically, the validated user identities include the user identities and values derived from the user keys. The validated user identities will be submitted with the request and used to demonstrate to the key distribution center 306 that the gaming system has knowledge of the user key and hence, implicitly authenticates the users.

One way to generate the key derivative value is to compute a cryptographic hash of the user key using the key of the game console. For user U_1 with key K_1, a hash H_1 is computed as follows:

$$H_1 = HMAC_{Kx}(K_1)$$

The hash H_1 forms the key derivative value. Another way is to encrypt the current time using the user key K_1, as follows:

$$H_1 = E_{K1}(T)$$

Once again, the resulting value H_1 forms the key derivative value. The validated user identity is the combination of the user identity U_1 and the corresponding key derivative value H_1:

$$\text{Validated User Identity} = (U_1, H_1).$$

At block 504 in Fig. 5, the game console 102 constructs a request containing the game console identity X, the game title identity G, the online service identity A of the service being requested, and multiple validated user identities (U_1, H_1), (U_2, H_2), (U_3, H_3), and (U_4, H_4). The request has the following identity string:

$$\text{Request} = [X, G, A, (U_1, H_1), (U_2, H_2), (U_3, H_3), (U_4, H_4)]$$

If the gaming system wanted authentication for more than online service, the identities of other services B, C, ..., etc. are added to the request. Additionally, the request may include a version of the authentication protocol and a random nonce generated by the game console to resist replay attacks. The request may further include a checksum value to be used to verify receipt of the entire identity string. The game console 102 submits the request over the network 302 to the key distribution center 306 (i.e., path 402).

At block 506 in Fig. 5, the key distribution center 306 evaluates the request as well as the identities contained in the request. The distribution center 306 generates a random session key to be used for each service being requested by the gaming system 100(1). In this example, the center 306 generates a random session

key K_{XA} to be used during the second communication cycle involving the game console 102 and the online service 304. If other services are requested, additional random session keys K_{XB}, K_{XC}, …, etc. are produced.

At block 508 in Fig. 5, the key distribution center 306 generates a ticket that will subsequently be presented to the requested online service. There is one ticket issued for each service requested, but each ticket is effective for multiple users. The ticket contains the identity string submitted in the request. It also includes a time T_G that the ticket is generated, a time T_L identifying the time length before expiration of the ticket, and the randomly generated session key K_{XA} for the service requested. The ticket contents are encrypted via a symmetric key cipher (e.g., DES) that utilizes the online service's key K_A, as follows:

$$\text{TicketA} = E_{KA}[T_G, T_L, K_{XA}, X, G, A, U_1, U_2, U_3, U_4]$$

Notice that the ticket does not carry the corresponding key derivative values H_i. Once the authentication server reads the key derivative values and believes the game console knows the user keys, the authentication server places the identities of the users within the issued tickets. Individual online services will subsequently believe in whatever the ticket tells it and hence do not need to see the key derivative values H_i.

If the gaming system sought tickets for more than one service, the key distribution center 306 issues multiple tickets, each ticket being encrypted with the public key of the desired online service, as follows:

$$\text{TicketB} = E_{KB}[T_G, T_L, K_{XB}, X, G, B, U_1, U_2, U_3, U_4]$$

$$\text{TicketC} = E_{Kc}[T_G, T_L, K_{XC}, X, G, C, U_1, U_2, U_3, U_4]$$

$$\vdots$$

$$\text{TicketN} = E_{KN}[T_G, T_L, K_{XN}, X, G, N, U_1, U_2, U_3, U_4]$$

At block 510 in Fig. 5, the key distribution center 306 returns each ticket over the network 302 to the gaming system 100(1) (i.e., path 404). Since the game console 102 does not know the online service's key K_A, the game console 102 cannot open the ticket and alter the contents. The key distribution center also returns the session keys in an attached encrypted message. The session key message contains the ticket generation time T_G, the ticket expiration length T_L, and one or more session keys K_{XA}, K_{XB}, K_{XC}, etc., and all contents are encrypted using the game console's key K_X, as follows:

$$\text{Session Key Message} = E_{KX}[T_G, T_L, K_{XA}, K_{XB}, K_{XC}, \ldots]$$

Since the session key message is encrypted with the game console's key K_X, the game console 102 is able to open the session key message and recover the session time parameters and session keys.

At block 512 in Fig. 5, the game console 102 passes the ticket, as is, onto the online service 304 (i.e., path 406). The game console 102 also generates and sends the current time T (instead of its own ID) encrypted with the random session key K_{XA}, as follows:

$$\text{Time Message} = E_{KXA}[T]$$

At block 514 in Fig. 5, the online service 304 evaluates the authenticity of the ticket. It decrypts the ticket using its key K_A to recover the contents, as follows:

$$D_{K_A}[\text{TicketA}] = T_G, T_L, K_{XA}, X, G, A, U_1, U_2, U_3, U_4$$

The decrypted contents include the session key K_{XA}. The online service then uses the session key K_{XA} to decrypt the time message and recover the time T, as follows:

$$D_{K_{XA}}[\text{Time Message}] = T$$

The online service 304 compares the recovered time sent from the game console with the current time. If the recovered time is not within an allowable time horizon from the current time, the online service deems the game console 102 as not authentic and the ticket as a forgery. In this case (i.e., the "No" branch from block 516), the online service denies service to the game console.

On the other hand, if the recovered time is within an allowable time window from the current time, the online service is assured that the game console 102 is authentic. In this case (i.e., the "Yes" branch from block 516), the online service generates a reply containing the time T in the request Time Message, encrypted with the session key K_{XA}, as follows:

$$\text{Reply} = E_{K_{XA}}[T]$$

At block 518, the online service 304 returns the reply over the network 302 to the gaming system 100(1) (i.e., path 408). Once again, this reply is piggybacked on a regular online service reply and does not incur an extra communication trip.

At block 520 in Fig. 5, the game console 102 evaluates the authenticity of the reply. The game console 102 decrypts the reply using the session key K_{XA} that it previously received in the session key message from the key distribution center 306. If the game console 102 can successfully decrypt the reply, and the T value is the same as the T value originally sent in the Time Message, the game console is assured that the online service is authentic because the online service could not otherwise have known the random session key K_{XA}. In this case (i.e., the "Yes" branch from block 522), the game console is free to interact with the services offered by the online service (block 524). Otherwise, if the reply cannot be decrypted successfully (i.e., the "No" branch from block 522), the game console deems the online service as not authentic and forgoes its services.

The multi-user authentication process is built atop the three-participant model in the Kerberos authentication process. It differs from Kerberos in that it allows authentication of multiple users simultaneously, with one trip to the key distribution center and one ticket for all users. The extension involved, in part, defining new message strings to be passed among the participants.

An alternative implementation is to leverage the existing well-defined data packets employed in the conventional Kerberos protocol. Kerberos defines a packet that includes a data field known as the "Pre-Auth" data field for pre-authorized data. In the alternate implementation, the multiple validated user identities (U_1, H_1), (U_2, H_2), (U_3, H_3), and (U_4, H_4) are inserted into this "Pre-

Auth" data field so that all users are authenticated in the same request. Also the Kerberos tickets issued would be modified to contain extra "Authorization Data" that contains the user identities that have been authenticated (U_1,U_2,U_3,U_4).

<u>Establishing Game Console Identity and Key</u>

The above multi-user authentication protocol hinges in part on the ability to establish the game console identity X and associated game console key K_X. This section describes how these parameters are created in the first place.

Generally, each game console 102 is manufactured with secret information that is very difficult to guess and very difficult to access. Tamper resistant designs are employed to physically protect the secret information on the game console from attacks on the hardware. The secret information is stored at a backend server for use later at account creation time to verify whether the game console is authentic. The secret information may be the same for all or a batch of consoles, or unique for each console. Preferably, the secret information is unique so that if the information is compromised, only a single fake account can be created.

The establishment of a secure game console identity relies on the fact that console manufacturing is a regulated process and that all consoles have consistent characteristics, regardless of the manufacturing date. It also relies on the fact that because manufacturing is controlled by one or a limited number of entities, a robust and secure database of the secret information unique to each machine can be generated and used on the backend to verify that a particular game console is, in fact, a real console at the time a game console account is being created.

Fig. 6 illustrates a process 600 for constructing a game console with secret information for use in establishing a console identity X and game console key K_X.

The process 600 is performed in part during manufacturing and in part during account creation in which an online gaming account for the game console is created. These two phases are distinguished in Fig. 6 by a horizontal dashed line. The tasks are illustrated beneath headings "Manufacturing", "Authentication Server" and "Game Console" to convey exemplary locations as to where the tasks are performed.

At block 602, the manufacturer constructs the game console to incorporate one or more pieces of information at the time of manufacture. The information is preferably of the type that can be made available programmatically. In one implementation, the game console is built to include at least two pieces of information, one that is unique to the individual machine and a second that is very difficult to guess and access. The first piece of information need not be evenly distributed across a name space, nor does it need to be secret or hard to read, but it does need to be unique across the name space and accessible programmatically. For example, the first piece of information could be a console serial number assigned by the manufacturer, a hard disk ID printed on the hard disk drive, or a serial number of any chip, ROM, firmware, or the like. This first piece of information will be used as the "name" of the game console to look up which row in table 650 corresponds to the particular game console.

The second piece of information does not necessarily need to be unique per machine, but is evenly distributed across its namespace. It is very difficult to guess and very difficult to access. Programmatic access is permitted via secure techniques, such as code-signing techniques, but physical access is controlled via hardware-based solutions that resist tampering and render physical attack and reverse engineering very difficult. Examples of the second piece of information

include a CPU ID or a value derived from the CPU ID, such as a one-way hash of the CPU ID. Another example is a random key written onto disk at manufacturing time, the disk drive needs to be protected from unauthorized reading (like using the CPU ID) through some other means. This second piece of information will be used as the "key" for the game console, to prove that the game console is genuine and not some other computing device pretending to be a game console.

At block 604, the information is recorded in a database at the manufacturer as a set of database records 650. Each record includes, for example, an identity of the console, the hard disk ID (HDID), and the CPU ID. At block 606, the database records 650 are securely made available to an authentication server that may or may not be hosted by the key distribution center 306. In one implementation, a secure database replication procedure is employed to securely transfer the console information database to the authentication server. It is further noted that the same entity may or may not control the manufacturing and authentication server.

At this point, the two pieces of information can be used in the verification of the game console and creation of the game console identity X and key K_X. Such information can be used to verify, for example, that the game console is valid and not an impersonating personal computer.

This completes the manufacturing phase of process 600. The game consoles are subsequently released from manufacturing and sold to consumers. When the user first decides to play an online game or access online services, the game console first obtains an account. This initiates the account creation phase of the process 600 in Fig. 6.

At block 610, the game console submits the information, or data derived from the information, to the authentication server. The information could be sent

over a secure link (e.g., SSL) established between the game console and the authentication server. In one implementation, the game console submits the hard disk ID and the CPU ID assuming a secure connection is established, such as using SSL to protect the contents of the message. To avoid exposing the CPU ID, another implementation is to submit in place of the CPU ID, a one-way hash digest of the CPU ID in such a way that it is extremely difficult to deduce the CPU ID from the resulting value. Another way to prove knowledge of the CPU ID and to also prevent replay attacks is to send $E_{CPUID}(T)$ where the current time is encrypted with the CPU ID as the Key.

At block 612, the authentication server evaluates the information from the game console. In one implementation, the authentication server uses the information as an index into the console database records 650 to identify that the information is a correct combination of pieces. For instance, the authentication server determines whether the hard disk ID and CPU ID passed in from the game console form a correct pair recorded in the database as belonging to the same game console. If there is a match, the authentication server can be assured that the client is a valid game console because it would be very difficult for an impostor to guess, manually or programmatically, both pieces of information and their relationship.

At block 614, the authentication server looks up in the console database records to determine whether an account has already been established for this information. If no account exists (i.e., the "No" branch from block 614), the authentication server creates an account and assigns a unique game console identity X and a randomly generated key K_X to that account (block 616). If an account exists (i.e., the "Yes" branch from block 614), the authentication server

retrieves the existing account information (block 618). The game console identity X and key K_X are returned to the game console (block 620).

At block 622, the game console stores the game console identity X and key K_X in a secure way to resist hardware attacks. It also sends back a message containing the identity X and the identity encrypted with the key K_X, or $E_{Kx}(X)$. Having the game console return a message allows the authentication server to ascertain whether the game console correctly received the account data. If the decrypted identity matches the non-encrypted identity, the game console received the account data correctly. If the two do not match, the game console received incorrect data and the authentication server should delete the account and restart the process at block 610. If no reply is received from the game console, the authentication server cannot be sure if the account data successfully made it to the game console. In this case, the authentication server flags the account and waits for another new account creation for this game console (signifying that the game console did not receive the account data) or a successful logon (indicating that the game console did receive the account data).

At game time, the actual security of the console machine is achieved via the identity X and key K_X, as described previously in the multi-user authentication process.

Conclusion

The above processes are advantageous for many reasons. First, the multi-user authentication process is fast because it accomplishes all authentication for the gaming system, including multiple users, the game title, and the machine itself, in a single roundtrip communication with the key distribution center. After this

initial roundtrip, the game console is able to communicate with any trusted services without re-communicating with the key distribution center. The whole process is accomplished with minimal user input and minimal delay before play can begin. The single ticket obtained from the authentication server is also unique in that it not only proves the particular game console, but also the multiple user identities playing at the game console. This is in contrast to the traditional approach where multiple tickets would need to be used (one for each authenticated principal).

Although the invention has been described in language specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claimed invention.